EXPlora-Lang

Aspirational Goal: make data analytics simple. For everyone.

Explicit Goal: automate constructing data pipelines and the visualization of results with the use of LLMs and significantly reduce the entry level of knowledge needed for the same or better quality results. We expect that with this language new analysts will be required to know less programming modules/libraries, compared to Python, as their features will be embedded into our language itself.

Research Questions and their Methodology:

1. **Question:** (Main Question) To what degree LLM outputs are aligned with humans' intent in the field of data analytics?

   Basic Idea: In our work, we plan to support the automatic generation of data pipelines: first as a more high-level execution plan, which is then transformed to certain code blocks in Python, which are then executed under the hood of our language. As LLMs reason well enough, but not perfectly, we want to minimize this discrepancy between the LLM plans and the actual intents of the users.

   We can measure the quality of alignment between LLM outputs and humans' intent in two following ways:

   - For each prompt we first collect a pair consisting of an LLM-generated plan (response to a task) and a corresponding plan created by human. Then with this paired dataset we can compare the difference between each two plans with a semantic similarity metric, such as BERTScore or BLEU. This approach is more accurate, but very time-consuming as we need to create many human-generated plans.
   - For each prompt, we first ask the LLM to generate a plan for the solution of the task and we ask another LLM (a judge) to check whether this plan solves the task given in the prompt or not. If the plan is correct, we give it score=1, else we give it score=0. Then we aggregate the scores for many tasks. This approach is not perfectly accurate, as LLMs can be sometimes biased, but this approach can be used as an alternative reasonable metric for measuring the quality of alignment to the task.

   As the second approach is way easier to implement and it corresponds to our research question well enough, we as a team will pick the second option.

Methodology:

- **Changes to a language:** JSON plan generator
- **What we plan to do:**
    - Compare LLM-built and human-defined plans using the approach described above
    - Improve the result. If we'll decide to use only one certain model, then we will use the techniques, like few-shot learning; if we'll decide to find the best model in terms of the quality of alignment with user's intent, then we'll just switch to another model
- **Resources:** compute resources (GPUs), LLM APIs, a small dataset of JSON plans (but probably we might not require it)
- **Why is it reliable:** by using the metric defined above, we can reasonably approximate the quality of the alignment. Ways of improvement are also quite simple and straight-forward and, as they usually help at improving scores in other tasks, we suppose they will help to improve our result here as well.

2. **Question:** To what degree can LLM improve efficiency of data analytics code in terms of memory and speed (compared to average Python data analyst)?

   Basic Idea: We would like to task the LLM to generate code which is similar or better than human-created one in terms of speed and memory consumption. To solve this problem, we plan to add a logger that will log how fast the code executes in terms of total execution time and how much additional memory is needed to solve the task. Then we plan to compare average times of execution of the code written by LLMs and the corresponding code created by humans (for each task we compare the means across 5 runs of the code), as well as average additional memory needed for the task. If the human solution is better than the LLM-generated one, then we tune our LLM with a human solution (using few-shot learning for example).

   Methodology:

   - **Changes to a language:** logger (time and additional memory analyzer)
   - **What we plan to do:** see above
   - **Resources:** compute resources (GPUs), LLM APIs, a small dataset of human-written and LLM-written code with given time and memory consumption
   - **Why is it reliable:** execution time and memory usage are objective, reproducible metrics that directly reflect performance. By having the LLM focus on generating this kind of output, we directly solve the problem, defined above.

3. **Question:** To what degree the graphs designed by LLMs are more preferred by consumers of analyzed information compared to human-designed graphs on average?

   Basic Idea: We want to fully delegate the visualization part to the LLM. However, we currently don't know whether LLM provides the graphical representations which the consumers will like. To answer this question, we will perform blind majority vote between a pair of graphs (where one is designed by LLM and another by human) on the selected audience.

   Methodology:

   - **Changes to a language:** LLM-powered function of visualization
   - **What we plan to do:**
     - Collect the dataset of pairs of specific graphs: ones built by LLMs and others designed by humans
     - Perform blind majority vote on several criteria, including:
       - ease of understanding of the graph
       - volume of represented info
       - beauty and tidiness of presentation
   - **Resources:** Audience, infrastructure for the survey (telegram bot)
   - **Why is it reliable:** Voting provides many insights into user preference**.** Therefore, we also may use it to answer our question and to improve the LLM results.

4. **Question:** How often do types inferred by EXPlora-Lang for LLM-generated code match the intended data types specified by humans?

   Basic Idea: The human defines what they expect (what types the code should use). The LLM generates code that implicitly assumes certain types. The Type Inference Engine (TIE) determines the actual types that the code implies when analyzed formally.

   The results are compared in pairs:

   a) Human / LLM : Does the LLM understand the user's intent?
   b) LLM / TIE: Did the LLM generate type-consistent code?
   c) Human / TIE: Does the resulting, type-checked program still express what the user meant?

   Methodology:

   - Dataset creation
     - The dataset should contain LLM generated outputs in response to human prompts that describe intended behavior and data types.
   - Annotation
     - Humans write down the intended types of variables / expressions / etc. for all prompts.
   - The TIE infers all types.
   - LLM-assumed types are extracted from explicit annotations or implicit cues in the generated code.
   - The three sets are compared in pairs as presented in the **Basic Idea** section above.
   - Computation of metrics
     - Things such as "Type Agreement Rate" or "Error Distance" are measured.
     - Metrics are used to improve LLM output.

   Reliability: Type mismatches can be caught by both humans and the TIE. Therefore, "alignment scores" can be given to the LLM based on how aligned the types are, with its goal being to receive higher scores.